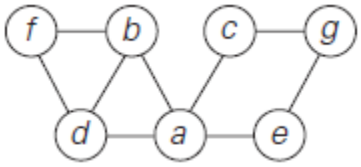
Tutorial 8

Graph Algorithms

CSC 311 (Fall 2018)

1- Consider the following graph.



a- Write down the adjacency matrix and adjacency lists specifying this graph. (Assume that the matrix rows and columns and vertices in the adjacency lists follow in the alphabetical order of the vertex labels.)

b- Starting at vertex *a* and resolving ties by the vertex alphabetical order, traverse the graph by breadth-first search and construct the corresponding depth-first search tree.

c- Starting at vertex *a* and resolving ties by the vertex alphabetical order, traverse the graph by depth-first search and construct the corresponding depth-first search tree. Give the order in which the vertices were reached for the first time (pushed onto the traversal stack) and the order in which the vertices became dead ends (popped off the stack).

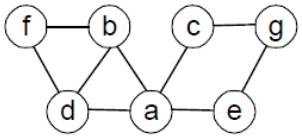
Solution:

a-

	a	*b*	*c*	*d*	*e*	*f*	*g*
a	0	1	1	1	1	0	0
b	1	0	0	1	0	1	0
c	1	0	0	0	0	0	1
d	1	1	0	0	0	1	0
e	1	0	0	0	0	0	1
f	0	1	0	1	0	0	0
g	0	0	1	0	1	0	0

a	$\rightarrow b \rightarrow c \rightarrow d \rightarrow e$
b	$\rightarrow a \rightarrow d \rightarrow f$
c	$\rightarrow a \rightarrow g$
d	$\rightarrow a \rightarrow b \rightarrow f$
e	$\rightarrow a \rightarrow g$
f	$\rightarrow b \rightarrow d$
g	$\rightarrow c \rightarrow e$

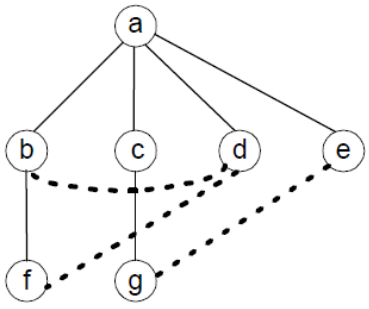
b- the graph; (ii) the traversal's queue; (iii) the tree (the tree and cross edges are shown with solid and dotted lines, respectively).



(i)

a b c d e f g

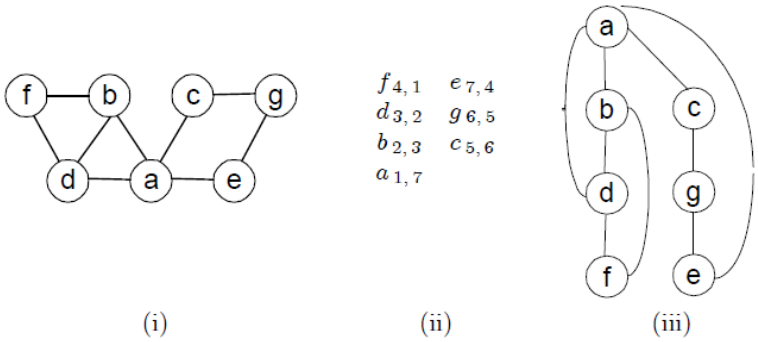
(ii)



(iii)

c- See below: (i) the graph; (ii) the traversal's stack (the first subscript number indicates the order in which the vertex was visited, i.e., pushed onto the stack, the second one

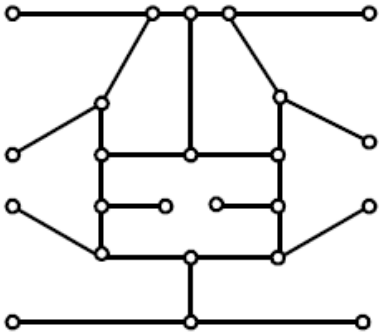
indicates the order in which it became a dead-end, i.e., popped off the stack); (iii) the DFS tree (with the tree edges shown with solid lines and the back edges shown with dashed lines).



(i) (ii) (iii)

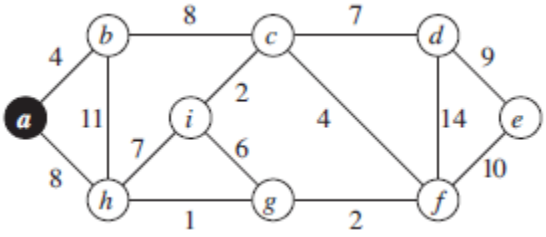
2- Construct such a graph for the following maze.

Which traversal—DFS or BFS—would you use if you found yourself in a maze and why?
Solution:



DFS is much more convenient for going through a maze than BFS. When DFS moves to a next vertex, it is connected to a current vertex by an edge (i.e., "close nearby" in the physical maze), which is not generally the case for BFS. In fact, DFS can be considered a generalization of an ancient right-hand rule for maze traversal: go through the maze in such a way so that your right hand is always touching a wall.

3- Given the following weighted graph, find a *spanning tree*



Solution: